UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/079,966	02/21/2002	Jeffrey R. Cobb	WILY-01008US0	9235

7590 09/08/2004

EXAMINER
BONZO, BRYCE P

ART UNIT	PAPER NUMBER
2114	

Burt Magen
Vierra Magen Marcus Harmon & DeNiro, LLP
685 Market Street, Suite 540
San Francisco, CA 94105-4206

DATE MAILED: 09/08/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/079,966	COBB ET AL.	
	Examiner	Art Unit	
	Bryce P Bonzo	2114	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *21 February 2002*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-79* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-9,12-21,29,33-36,38-42,44-47 and 52-77* is/are rejected.

7)☒ Claim(s) *10,11,22-28,30-32,37,43,48-51,70,71,78 and 79* is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *21 February 2002* is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

NON-FINAL OFFICIAL ACTION

Status of the Claims

Claims 52-57 and 64-71 are rejected under 35 USC §101.

Claims 1-3, 13-17, 29, 33-35, 38-41, 44, 52-55, 57-61, 63-67, 72-75 are rejected under 35 USC §102.

Claims 4-9, 12, 18-21, 36, 42, 45-47, 56, 62, 68, 69, 76 and 77 are rejected under 35 USC §103.

Claims 10, 11, 22-28, 30-32, 37, 43, 48-51, 70, 71, 78, 79 are objected to while containing allowable subject matter.

Claim 39 is objected to based on a minor typographical error.

Objections

Claim 39 recites "whether said method thread." The Examiner notes to maintain consistency within the claim and with claim 33, the instance of thread should be removed. Appropriate action is required.

Rejections under 35 USC §101

35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

Claims 52-57 and 64-71 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. These claims fail to

claim any active steps carried out by a processor, and as such are rejected as being a computer program per se. Application is advised to provide a limitation requiring the execution of code by the processor to carry out the prescribed steps. The Examiner recommends the following modification to the claims which will obviate the rejection:

> One or more processor readable storage devices having processor readable code embodied on said processor readable storage devices, said processor readable code for programming one or more processors which when executing the code performs ~~to perform~~ a method comprising the steps of:

Rejections under 35 USC §102

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

Claims 1-3, 13-17, 29, 33-35, 38-41, 44, 52-55, 57-61, 63-67, 72-75 are rejected under 35 U.S.C. 102(e) as being anticipated by Os (United States

Patent Application Publication US 2002/0162053 A1). The Examiner points out that Os is the child of application 09/265,839 filed March 10th, 1999.

As per claim 1, Os discloses:

A method for detecting whether a routine has stalled, comprising the steps of:

accessing existing code for a first routine (Page 2, ¶18);

automatically modifying said existing code to include new code (page 2, ¶18); and

using said new code to determine if said first routine has stalled (page 1, ¶10).

As per claim 2, Os discloses:

said existing code is object code (pages 1 and 2, ¶17); and

said new code is object code (page 2, ¶18: .exe files are object code).

As per claim 3, Os discloses:

further comprising the step of: receiving a rule, said rule identifies said first routine and an interval, said step of automatically modifying is performed in response to said rule (page 2, ¶18), said first routine is considered to be stalled after a determination that said first routine has been running for at least as long as said interval (page 2, ¶27).

As per claim 13, Os discloses:

comprising the steps of: receiving an indication that a particular routine is running, said particular routine is one of a plurality of routines that comprise a process (page 2, ¶18); and

automatically determining whether said particular routine has stalled (page 2, ¶18).

As per claim 14, Os discloses:

said particular routine is a thread (Os discloses the use of .dll files a hallmark of object oriented programming in which threads and methods act on objects).

As per claim 15, Os discloses:

said plurality of routines are threads (page 1, ¶17);

at least two or more of said threads, including said particular routine, are run concurrently (page 1, 2 ¶17,18 Os discloses the use of .dll files a hallmark of object oriented programming in which threads and methods act on objects).

As per claim 16, Os discloses:

said particular routine is a method (Os discloses the use of .dll files a hallmark of object oriented programming in which threads and methods act on objects).

As per claim 17, Os discloses:

comprising the step of: receiving an indication of said method, said step of determining whether said particular routine is stalled includes detecting a situation when a thread enters said method and does not return within an approximation of an expected time frame (column 2, ¶23).

As per claim 29, Os discloses:

further comprising the step of: reporting said particular routine as being stalled if said particular routine was determined to be stalled (page 2, ¶28).

As per claim 33, Os discloses:

receiving an indication that a particular thread is running (page 2, ¶18); and

determining whether said particular thread has stalled (page 2, ¶27).

As per claim 34, Os discloses:

receiving an indication of a first method, said step of determining whether said particular thread is stalled includes detecting a situation when said thread enters said first method and does not return within an approximation of an expected time frame (page 2, ¶27).

As per claim 35, Os discloses:

automatically modifying existing object code in order to add new object code, said new object code performs said step of determining (page 2, ¶18).

As per claim 38, Os discloses:

said particular thread is one of multiple threads running concurrently and which comprise a process (page 2, ¶18).

As per claim 39, Os discloses:

receiving an indication that a particular method is running (page 2, ¶18); and

determining whether said method thread has stalled (page 2, ¶27).

As per claim 40, Os discloses:

receiving an indication that identifies said particular method from a set of methods, said step of determining whether said particular method is stalled includes detecting a situation when a thread enters said particular method and does not return within an approximation of an expected time frame (column 2, ¶27).

As per claim 41, Os discloses:

automatically modifying existing object code for said particular method in order to add new object code, said new object code performs said step of determining (page 2, ¶18).

As per claim 44, Os discloses:

wherein said particular method is one of multiple methods running concurrently and which comprise a process (page 2, ¶18).

As per claim 52, Os discloses:

accessing existing code for a first routine (page 2, ¶18);

automatically modifying said existing code to include new code (page 2, ¶18); and

using said new code to determine if said first routine has stalled (page 2, ¶18).

As per claim 53, Os discloses:

said existing code is object code (pages 1 and 2, ¶17); and

said new code is object code (page 2, ¶18: .exe files are object code).

As per claim 54, Os discloses:

wherein said method further comprises the step of: receiving a rule, said rule identifies said first routine and an interval, said step of automatically

modifying is performed in response to said rule, said first routine is considered to be stalled after a determination that said first routine has been running for at least as long as said interval (page 2, ¶18, 22, 27).

As per claim 55, Os discloses:

said first routine is one of a plurality of routines that comprise a process (column 2, ¶18).

As per claim 57, Os discloses:

said first routine is a thread performing a method (Os discloses the use of .dll files a hallmark of object oriented programming in which threads and methods act on objects); and

said step of using includes determining whether said thread entered said method and did not return within a predetermined time period (page 2, ¶27).

As per claims 58, Os discloses:

one or more storage devices (inherent to a computer); and

one or more processors in communication with said one or more storage devices (inherent to a computer), said one or more processors perform a method comprising the steps of:

accessing existing code for a first routine (page 2, ¶18),

automatically modifying said existing code to include new code (page 2, ¶18), and

using said new code to determine if said first routine has stalled (page 2, ¶18).

As per claim 59, Os discloses:

said existing code is object code (pages 1 and 2, ¶17); and

said new code is object code (page 2, ¶18: .exe files are object code).

As per claim 60, Os discloses:

wherein said method further comprises the step of: receiving a rule, said rule identifies said first routine and an interval, said step of automatically modifying is performed in response to said rule, said first routine is considered to be stalled after a determination that said first routine has been running for at least as long as said interval (page 2, ¶18, 22, 27).

As per claim 61, Os discloses:

said first routine is one of a plurality of routines that comprise a process (column 2, ¶18).

As per claim 63, Os discloses:

said first routine is a thread performing a method (Os discloses the use of .dll files a hallmark of object oriented programming in which threads and methods act on objects); and

said step of using includes determining whether said thread entered said method and did not return within a predetermined time period (page 2, ¶27).

As per claim 64, Os discloses:

receiving an indication that a particular routine is running, said particular routine is one of a plurality of routines that comprise a process (page 2, ¶18); and

automatically determining whether said particular routine has stalled (page 2, ¶18).

As per claim 65, Os discloses:

said particular routine is a thread (Os discloses the use of .dll files a hallmark of object oriented programming in which threads and methods act on objects).

As per claim 66, Os discloses:

said particular routine is a method (Os discloses the use of .dll files a hallmark of object oriented programming in which threads and methods act on objects).

As per claim 67, Os discloses:

receiving an indication of said method, said step of determining whether said particular routine is stalled includes detecting a situation when a thread

enters said method and does not return within an approximation of an expected time frame (page 2, ¶18, ¶27).

As per claim 72, Os discloses:

one or more storage devices (inherent to a computer system); and

one or more processors in communication with said one or more storage devices (inherent to a computer system), said one or more processors perform a method comprising the steps of:

receiving an indication that a particular routine is running, said particular routine is one of a plurality of routines that comprise a process (page 2, ¶18), and

automatically determining whether said particular routine has stalled (page 2, ¶18).

As per claim 73, Os discloses:

said particular routine is a thread (Os discloses the use of .dll files a hallmark of object oriented programming in which threads and methods act on objects).

As per claim 74, Os discloses:

said particular routine is a method (Os discloses the use of .dll files a hallmark of object oriented programming in which threads and methods act on objects).

As per claim 75, Os discloses:

receiving an indication of said method, said step of determining whether said particular routine is stalled includes detecting a situation when a thread enters said method and does not return within an approximation of an expected time frame (page 2, ¶18).

Rejections under 35 USC §103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Claims 4-9, 12, 18-21, 36, 42, 45-47, 56, 62, 68, 69, 76 and 77 rejected under 35 U.S.C. 103(a) as being unpatentable over Os (United States Patent Application Publication US 2002/0162053 A1). The Examiner points out that Os is the child of application 09/265,839 filed March 10th, 1999.

As per claim 4, Os discloses:

adding code for a timing mechanism to said existing code;

adding code for starting said timing mechanism to said existing code;

adding a first instruction to said first routine, said first instruction causes the execution of said code for starting said timing mechanism (all of the above are disclosed at page 2, ¶27).

Os does not explicitly disclose:

adding code for stopping said timing mechanism to said existing code;

adding a second instruction to said first routine, said second instruction causes the execution of said code for stopping said timing mechanism.

Official Notice is given that was well known at the time of invention by computer programmers provide instructions to stop timers and carry out the stopping of timers in monitoring systems in timeout monitoring systems. Providing these instructions to stop timers is crucial in timeout based systems. This is because if the timer is not stopped, it will generate unwarranted timeout error messages after what ever it is monitoring has successfully completed, causing a system to report or take action for conditions which are not present. Thus it would have been obvious to one of ordinary skill in the art at the time of invention to incorporate the practice and instructions for stopping timers into the system OS thereby creating a system which is less likely to create false error messages after a possible error causing occurrence has successfully completed.

As per claim 5, Os does not explicitly disclose:

said second instruction is added such that it is executed at all exits of said routine. The Examiner takes Official Notice that was well held by the computer programming community prior to the time of invention to clean up processes and

objects instantiated by a routine when the routine is terminated. This practice encompasses the concepts of garbage collection and memory leaking. If a programmer does not carry out this cleaning up of memory spaces and child processes after the parent routine has completed, the left over data and processes may interfere with the processes which remain active. C++ programmers are explicitly taught to clean up the memory themselves in the program, while Java programmers are free from this burden as the programming language and complier are designed to automatically carry out these procedures. Thus it would have been obvious to one of ordinary skill in the art of computer programming to stop any timers for monitoring a routine which were created by the routine when the routine exits, thus preventing memory leakage and garbage build up in the memory space.

As per claim 9, Os discloses:

receiving an indication that said first routine has started (page 2, ¶18);

starting a timing mechanism in response to said step of receiving (page 2, ¶18);

reporting said first routine as stalled if said timing mechanism is not stopped prior to a determination that said timing mechanism is overdue (page 2, ¶27).

Os does not explicitly disclose:

receiving an indication that said first routine has completed, if said first routine has completed;

stopping said timing mechanism in response to receiving said indication that said first routine has completed;

Official Notice is given that was well known at the time of invention by computer programmers provide instructions to stop timers and carry out the stopping of timers in monitoring systems in timeout monitoring systems. Providing these instructions to stop timers is crucial in timeout based systems. This is because if the timer is not stopped, it will generate unwarranted timeout error messages after what ever it is monitoring has successfully completed, causing a system to report or take action for conditions which are not present. Thus it would have been obvious to one of ordinary skill in the art at the time of invention to incorporate the practice and instructions for stopping timers into the system OS thereby creating a system which is less likely to create false error messages after a possible error causing occurrence has successfully completed.

As per claim 12, Os does not explicitly disclose:

said first routine is a thread executing a method, said indication that said first routine has completed indicates that said thread has exited said method. The Examiner takes Official Notice that was well held by the computer programming community prior to the time of invention to clean up processes and objects instantiated by a routine when the routine is terminated. This practice encompasses the concepts of garbage collection and memory leaking. If a programmer does not carry out this cleaning up of memory spaces and child processes after the parent routine has completed, the left over data and

processes may interfere with the processes which remain active. C++ programmers are explicitly taught to clean up the memory themselves in the program, while Java programmers are free from this burden as the programming language and complier are designed to automatically carry out these procedures. Thus it would have been obvious to one of ordinary skill in the art of computer programming to stop any timers for monitoring a routine which were created by the routine when the routine exits, thus preventing memory leakage and garbage build up in the memory space.

As per claim 18, Os discloses:

said indication that a particular routine is running is an indication that said particular routine has started (page 2, ¶18)

receiving an indication that said first routine has started (page 2, ¶18);

starting a timing mechanism in response to said step of receiving (page 2, ¶18);

reporting said first routine as stalled if said timing mechanism is not stopped prior to a determination that said timing mechanism is overdue (page 2, ¶27).

Os does not explicitly disclose:

receiving an indication that said first routine has completed, if said first routine has completed;

stopping said timing mechanism in response to receiving said indication that said first routine has completed;

Official Notice is given that was well known at the time of invention by computer programmers provide instructions to stop timers and carry out the stopping of timers in monitoring systems in timeout monitoring systems. Providing these instructions to stop timers is crucial in timeout based systems. This is because if the timer is not stopped, it will generate unwarranted timeout error messages after what ever it is monitoring has successfully completed, causing a system to report or take action for conditions which are not present. Thus it would have been obvious to one of ordinary skill in the art at the time of invention to incorporate the practice and instructions for stopping timers into the system OS thereby creating a system which is less likely to create false error messages after a possible error causing occurrence has successfully completed.

As per claim 20, Os explicitly discloses:

automatically adding new code to existing code for said particular routine, said new code performs said step of starting a timing mechanism, and reporting (page 2, ¶27).

Os does not explicitly disclose:

automatically adding new code to existing code for said particular routine, said new code performs said step of starting a timing mechanism, *stopping said timing mechanism* and reporting

Official Notice is given that was well known at the time of invention by computer programmers provide instructions to stop timers and carry out the stopping of timers in monitoring systems in timeout monitoring systems.

Providing these instructions to stop timers is crucial in timeout based systems. This is because if the timer is not stopped, it will generate unwarranted timeout error messages after what ever it is monitoring has successfully completed, causing a system to report or take action for conditions which are not present. Thus it would have been obvious to one of ordinary skill in the art at the time of invention to incorporate the practice and instructions for stopping timers into the system OS thereby creating a system which is less likely to create false error messages after a possible error causing occurrence has successfully completed.

As per claim 21, Os discloses:

automatically modifying existing object code for said particular routine in order to add new object code to said existing object code for said particular routine, said new object code performs said step of starting a timing mechanism, and reporting (page 2, ¶27).

Os does not explicitly disclose:

automatically modifying existing object code for said particular routine in order to add new object code to said existing object code for said particular routine, said new object code performs said step of starting a timing mechanism, *stopping said timing mechanism* and reporting.

Official Notice is given that was well known at the time of invention by computer programmers provide instructions to stop timers and carry out the stopping of timers in monitoring systems in timeout monitoring systems. Providing these instructions to stop timers is crucial in timeout based systems.

This is because if the timer is not stopped, it will generate unwarranted timeout error messages after what ever it is monitoring has successfully completed, causing a system to report or take action for conditions which are not present. Thus it would have been obvious to one of ordinary skill in the art at the time of invention to incorporate the practice and instructions for stopping timers into the system OS thereby creating a system which is less likely to create false error messages after a possible error causing occurrence has successfully completed.

As per claim 36, Os discloses:

starting a timing mechanism in response to said step of receiving an indication that said particular thread has started (page 2, ¶18),

reporting said particular thread as stalled if said timing mechanism is not stopped prior to a determination that said timing mechanism is overdue (page 2, ¶18)

Os does not explicitly disclose:

receiving an indication that said particular thread has completed, if said particular thread has completed,

stopping said timing mechanism in response to receiving said indication that said particular thread has completed

Official Notice is given that was well known at the time of invention by computer programmers provide instructions to stop timers and carry out the stopping of timers in monitoring systems in timeout monitoring systems. Providing these instructions to stop timers is crucial in timeout based systems.

This is because if the timer is not stopped, it will generate unwarranted timeout error messages after what ever it is monitoring has successfully completed, causing a system to report or take action for conditions which are not present. Thus it would have been obvious to one of ordinary skill in the art at the time of invention to incorporate the practice and instructions for stopping timers into the system OS thereby creating a system which is less likely to create false error messages after a possible error causing occurrence has successfully completed.

As per claim 42, Os discloses:

starting a timing mechanism in response to said step of receiving an indication that said particular method has started (page 2, ¶18),

reporting said particular method as stalled if said timing mechanism is not stopped prior to a determination that said timing mechanism is overdue (page 2, ¶18)

Os does not explicitly disclose:

receiving an indication that said particular method has completed, if said particular thread has completed,

stopping said timing mechanism in response to receiving said indication that said particular method has completed.

Official Notice is given that was well known at the time of invention by computer programmers provide instructions to stop timers and carry out the stopping of timers in monitoring systems in timeout monitoring systems. Providing these instructions to stop timers is crucial in timeout based systems.

This is because if the timer is not stopped, it will generate unwarranted timeout error messages after what ever it is monitoring has successfully completed, causing a system to report or take action for conditions which are not present. Thus it would have been obvious to one of ordinary skill in the art at the time of invention to incorporate the practice and instructions for stopping timers into the system OS thereby creating a system which is less likely to create false error messages after a possible error causing occurrence has successfully completed.

As per claim 45, Os discloses:

receiving an indication that a first routine has started (page 2, ¶18);

starting a timing mechanism in response to said indication that said first routine has started (page 2, ¶18);

reporting said first routine as stalled if said timing mechanism is not stopped prior to a determination that said timing mechanism is overdue (page 2, ¶27).

Os does not explicitly disclose:

receiving an indication that said first routine has completed, if said first routine has completed;

stopping said timing mechanism in response to receiving said indication that said first routine has completed.

Official Notice is given that was well known at the time of invention by computer programmers provide instructions to stop timers and carry out the stopping of timers in monitoring systems in timeout monitoring systems.

Providing these instructions to stop timers is crucial in timeout based systems. This is because if the timer is not stopped, it will generate unwarranted timeout error messages after what ever it is monitoring has successfully completed, causing a system to report or take action for conditions which are not present. Thus it would have been obvious to one of ordinary skill in the art at the time of invention to incorporate the practice and instructions for stopping timers into the system OS thereby creating a system which is less likely to create false error messages after a possible error causing occurrence has successfully completed.

As per claim 46, Os discloses:

automatically adding new code to existing code for said first routine, said new code performs said step of starting a timing mechanism, stopping said timing mechanism and reporting (page 2, ¶18).

As per claim 47, Os discloses:

automatically modifying existing object code for said particular routine in order to add new object code to said existing object code for said particular routine, said new object code performs said step of starting a timing mechanism, and reporting (page 2, ¶27).

Os does not explicitly disclose:

automatically modifying existing object code for said particular routine in order to add new object code to said existing object code for said particular

routine, said new object code performs said step of starting a timing mechanism, *stopping said timing mechanism* and reporting.

Official Notice is given that was well known at the time of invention by computer programmers provide instructions to stop timers and carry out the stopping of timers in monitoring systems in timeout monitoring systems. Providing these instructions to stop timers is crucial in timeout based systems. This is because if the timer is not stopped, it will generate unwarranted timeout error messages after what ever it is monitoring has successfully completed, causing a system to report or take action for conditions which are not present. Thus it would have been obvious to one of ordinary skill in the art at the time of invention to incorporate the practice and instructions for stopping timers into the system OS thereby creating a system which is less likely to create false error messages after a possible error causing occurrence has successfully completed.

As per claim 56, Os discloses:

receiving an indication that said first routine has started (page 2, ¶18);

starting a timing mechanism in response to said step of receiving (page 2, ¶18);

reporting said first routine as stalled if said timing mechanism is not stopped prior to a determination that said timing mechanism is overdue (page 2, ¶27).

Os does not explicitly disclose:

receiving an indication that said first routine has completed, if said first routine has completed;

stopping said timing mechanism in response to receiving said indication that said first routine has completed;

Official Notice is given that was well known at the time of invention by computer programmers provide instructions to stop timers and carry out the stopping of timers in monitoring systems in timeout monitoring systems. Providing these instructions to stop timers is crucial in timeout based systems. This is because if the timer is not stopped, it will generate unwarranted timeout error messages after what ever it is monitoring has successfully completed, causing a system to report or take action for conditions which are not present. Thus it would have been obvious to one of ordinary skill in the art at the time of invention to incorporate the practice and instructions for stopping timers into the system OS thereby creating a system which is less likely to create false error messages after a possible error causing occurrence has successfully completed.

As per claim 62, Os discloses:

receiving an indication that said first routine has started (page 2, ¶18);

starting a timing mechanism in response to said step of receiving (page 2, ¶18);

reporting said first routine as stalled if said timing mechanism is not stopped prior to a determination that said timing mechanism is overdue (page 2, ¶27).

Os does not explicitly disclose:

receiving an indication that said first routine has completed, if said first routine has completed;

stopping said timing mechanism in response to receiving said indication that said first routine has completed;

Official Notice is given that was well known at the time of invention by computer programmers provide instructions to stop timers and carry out the stopping of timers in monitoring systems in timeout monitoring systems. Providing these instructions to stop timers is crucial in timeout based systems. This is because if the timer is not stopped, it will generate unwarranted timeout error messages after what ever it is monitoring has successfully completed, causing a system to report or take action for conditions which are not present. Thus it would have been obvious to one of ordinary skill in the art at the time of invention to incorporate the practice and instructions for stopping timers into the system OS thereby creating a system which is less likely to create false error messages after a possible error causing occurrence has successfully completed.

As per claim 68, Os discloses:

receiving an indication that said first routine has started (page 2, ¶18);

starting a timing mechanism in response to said step of receiving (page 2, ¶18);

reporting said first routine as stalled if said timing mechanism is not stopped prior to a determination that said timing mechanism is overdue (page 2, ¶27).

Os does not explicitly disclose:

receiving an indication that said first routine has completed, if said first routine has completed;

stopping said timing mechanism in response to receiving said indication that said first routine has completed;

Official Notice is given that was well known at the time of invention by computer programmers provide instructions to stop timers and carry out the stopping of timers in monitoring systems in timeout monitoring systems. Providing these instructions to stop timers is crucial in timeout based systems. This is because if the timer is not stopped, it will generate unwarranted timeout error messages after what ever it is monitoring has successfully completed, causing a system to report or take action for conditions which are not present. Thus it would have been obvious to one of ordinary skill in the art at the time of invention to incorporate the practice and instructions for stopping timers into the system OS thereby creating a system which is less likely to create false error messages after a possible error causing occurrence has successfully completed.

As per claim 69, Os does not explicitly discloses:

said first routine is a thread executing a method, said indication that said first routine has completed indicates that said thread has exited said method.

The Examiner takes Official Notice that was well held by the computer programming community prior to the time of invention to clean up processes and objects instantiated by a routine when the routine is terminated. This practice encompasses the concepts of garbage collection and memory leaking. If a programmer does not carry out this cleaning up of memory spaces and child processes after the parent routine has completed, the left over data and processes may interfere with the processes which remain active. C++ programmers are explicitly taught to clean up the memory themselves in the program, while Java programmers are free from this burden as the programming language and complier are designed to automatically carry out these procedures. Thus it would have been obvious to one of ordinary skill in the art of computer programming to stop any timers for monitoring a routine which were created by the routine when the routine exits, thus preventing memory leakage and garbage build up in the memory space.

As per claim 76, Os discloses:

receiving an indication that said first routine has started (page 2, ¶18);

starting a timing mechanism in response to said step of receiving (page 2, ¶18);

reporting said first routine as stalled if said timing mechanism is not stopped prior to a determination that said timing mechanism is overdue (page 2, ¶27).

Os does not explicitly disclose:

receiving an indication that said first routine has completed, if said first routine has completed;

stopping said timing mechanism in response to receiving said indication that said first routine has completed;

Official Notice is given that was well known at the time of invention by computer programmers provide instructions to stop timers and carry out the stopping of timers in monitoring systems in timeout monitoring systems. Providing these instructions to stop timers is crucial in timeout based systems. This is because if the timer is not stopped, it will generate unwarranted timeout error messages after what ever it is monitoring has successfully completed, causing a system to report or take action for conditions which are not present. Thus it would have been obvious to one of ordinary skill in the art at the time of invention to incorporate the practice and instructions for stopping timers into the system OS thereby creating a system which is less likely to create false error messages after a possible error causing occurrence has successfully completed.

As per claim 77, Os discloses:

automatically modifying existing object code for said particular routine in order to add new object code to said existing object code for said particular routine, said new object code performs said step of starting a timing mechanism, and reporting (page 2, ¶27).

Os does not explicitly disclose:

automatically modifying existing object code for said particular routine in order to add new object code to said existing object code for said particular routine, said new object code performs said step of starting a timing mechanism, *stopping said timing mechanism* and reporting.

Official Notice is given that was well known at the time of invention by computer programmers provide instructions to stop timers and carry out the stopping of timers in monitoring systems in timeout monitoring systems. Providing these instructions to stop timers is crucial in timeout based systems. This is because if the timer is not stopped, it will generate unwarranted timeout error messages after what ever it is monitoring has successfully completed, causing a system to report or take action for conditions which are not present. Thus it would have been obvious to one of ordinary skill in the art at the time of invention to incorporate the practice and instructions for stopping timers into the system OS thereby creating a system which is less likely to create false error messages after a possible error causing occurrence has successfully completed.

Allowable Matter

Claims 10, 11, 22-28, 30-32, 37, 43, 48-51, 78 and 79 objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims.

Claims 70 and 71 are objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in independent form including all

of the limitations of the base claim and any intervening claims and the claims are amended to overcome the rejections under 35 USC §101.

The following is a statement of reasons for the indication of allowable subject matter. The following portions in combination with the remainder of the claims overcome the prior art.

As per claim 10:

A method according to claim 9, wherein said step of using further comprises the steps of: accessing a current time; verifying that said first routine is not known to be stalled or completed; accessing said due time; and determining whether first due time is earlier than said current time, said timing mechanism is overdue if said step of determining concludes that said first due time is earlier than said current time.

As per claim 11:

A method according to claim 9, wherein said step of stopping said timing mechanism comprises the steps of: determining whether said first routine has been reported as being stalled; changing said reporting to no longer indicate that said first routine is stalled if said step of determining concludes that said first routine has been reported as being stalled; and stopping said timing mechanism if said first routine has not been reported as being stalled

As per claim 22:

A method according to claim 18, wherein said step of determining whether said particular routine has started further comprises the steps of: accessing a current time; verifying that said particular routine is not known to be stalled or completed; accessing a due time; and determining whether due time is earlier than said current time, said timing mechanism is overdue if it is determined that said due time is earlier than said current time.

As per claim 23:

A method according to claim 18, wherein said step of stopping said timing mechanism comprises the steps of: determining whether said particular routine has been reported as being stalled; changing said reporting to no longer indicate that said particular routine is stalled if it is determined that said particular routine has been reported as being stalled; and stopping said timing mechanism if said particular routine has not been reported as being stalled.

As per claims 24-28:

A method according to claim 18, wherein said step of starting said timing mechanism comprises the steps of: receiving a threshold; accessing a current time; determining a first due time based on said threshold and said current time; and adding a indication of said particular routine and said first due time to a set of due times for other routines, said timing mechanism is overdue after said timing mechanism determines that said due time has been exceeded.

As per claim 30:

A method according to claim 29, wherein: said particular routine is an instance of a defined routine; and said step of reporting includes incrementing a counter that represents a number of instances of said defined routine that are currently stalled and reporting said number of instances of said defined routine that are currently stalled.

As per claim 31:

A method according to claim 29, wherein: said particular routine is an instance of a defined routine; and said step of reporting includes determining and reporting how many instances of said defined routine were stalled at a specified time.

As per claim 32:

A method according to claim 29, wherein: said particular routine is an instance of a defined routine; and said step of reporting includes receiving a customizable specified time period and reporting how many instances of said defined routine were stalled during said specified time period.

As per claim 37:

A method according to claim 36, wherein said step of stopping said timing mechanism comprises the steps of: determining whether said particular routine has been reported as being stalled; changing said reporting to no longer indicate

that said particular routine is stalled if it is determined that said particular routine has been reported as being stalled; and stopping said timing mechanism if said particular routine has not been reported as being stalled.

As per claim 43:

A method according to claim 42, wherein said step of stopping said timing mechanism comprises the steps of: determining whether said particular method has been reported as being stalled; changing said reporting to no longer indicate that said particular method is stalled if it is determined that said particular method has been reported as being stalled; and stopping said timing mechanism if said particular method has not been reported as being stalled.

As per claim 48:

A method according to claim 45, further comprising the steps of: accessing a current time; verifying that said first routine is not known to be stalled or completed; accessing a due time for said first routine; and determining whether said due time is earlier than said current time, said timing mechanism is overdue if said step of determining concludes that said first due time is earlier than said current time.

As per claim 49:

A method according to claim 45, wherein said step of stopping said timing mechanism comprises the steps of: determining whether said first routine has

been reported as being stalled; changing said reporting to no longer indicate that said first routine is stalled if said first routine has been reported as being stalled; and stopping said timing mechanism if said first routine has not been reported as being stalled.

As per claim 50:

A method according to claim 45, wherein: said first routine is an instance of a defined routine; and said step of reporting includes incrementing a counter that represents a number of instances of said defined routine that are stalled and reporting said number of instances of said defined routine that are stalled.

As per claim 51:

A method according to claim 45, wherein: said first routine is an instance of a defined routine; said step of reporting includes receiving a customizable specified time period and reporting how many instances of said defined were stalled during said specified time period.

As per claim 70:

One or more processor readable storage devices according to claim 68, wherein said step of stopping said timing mechanism comprises the steps of: determining whether said particular routine has been reported as being stalled; changing said reporting to no longer indicate that said particular routine is stalled if it is determined that said particular routine has been reported as being stalled;

and stopping said timing mechanism if said particular routine has not been reported as being stalled.

As per claim 71:

One or more processor readable storage devices according to claim 64, wherein: said particular routine is an instance of a defined routine; and said step of automatically determining includes incrementing a counter that represents a number of instances of said defined routine that are currently stalled and reporting said number of instances of said defined routine that are currently stalled.

As per claim 78:

An apparatus according to claim 72, wherein said step of stopping said timing mechanism comprises the steps of: determining whether said particular routine has been reported as being stalled; changing said reporting to no longer indicate that said particular routine is stalled if it is determined that said particular routine has been reported as being stalled; and stopping said timing mechanism if said particular routine has not been reported as being stalled.

As per claim 79:

An apparatus according to claim 72, wherein: said particular routine is an instance of a defined routine; and said step of automatically determining includes incrementing a counter that represents a number of instances of said defined

routine that are currently stalled and reporting said number of instances of said defined routine that are currently stalled.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Bryce P Bonzo whose telephone number is (703) 305-4834 or upon moving to the new facilities in Alexandria (571) 272-3655. The examiner can normally be reached on Monday-Friday.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Robert Beausoliel can be reached on (703) 305-9713 or upon moving to the new facilities in Alexandria (571) 272-3645. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Bryce P. Bonzo

Bryce P Bonzo
Examiner
Art Unit 2114